AD-A032 324 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 9/2
THE DESIGN AND IMPLEMENTATION OF A GENERAL PURPOSE INTERACTIVE --ETC(U)
SEP 76 B J STANKOWSKI

UNCLASSIFIED NL

1 OF 2
ADA032324

AD A032324

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DDC
RECEIVED
NOV 23 1976
B

THESIS

THE DESIGN AND IMPLEMENTATION
OF A
GENERAL PURPOSE INTERACTIVE GRAPHICS
SUBROUTINE LIBRARY

by

Barbara Jo Stankowski

September 1976

Thesis Advisor: Gary M. Raetz

Approved for public release; distribution unlimited.

COPY AVAILABLE TO DDC DOES NOT
PERMIT FULLY LEGIBLE PRODUCTION

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Design and Implementation of a General Purpose Interactive Graphics Subroutine Library.		5. TYPE OF REPORT & PERIOD COVERED Master's thesis; September 1976
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Barbara Jo Stankowski		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1976
		13. NUMBER OF PAGES 124
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

graphics
high level interface

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis describes the design and implementation of a high level, general purpose, interactive graphics subroutine library for the Vector General Interactive Graphics Display System. The lower interface levels of the system and its effects on the development of this high level interface are discussed. The problems and various approaches associated with the development of a general purpose, high level

(next page)

DD FORM 1 JAN 73 1473 (Page 1) EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

251 450 AB

20. (cont.)

applications subroutine libary that is user oriented is outlined. The basic design goals, solutions and recommendations for further expansion of the system are presented. This graphics subroutine library is implemented within the conventions of the C-Programming language and the UNIX operating system as implemented on the PDP-11/50 in the Naval Postgraduate School Computer Laboratory.

DD Form 1473
1 Jan 73
S/N 0102-014-6601

The Design and Implementation
of a
General Purpose Interactive Graphics
Subroutine Library

by

Barbara Jo Stankowski
Lieutenant, United States Navy
B.S., Pennsylvania State University, 1970

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
SEPTEMBER 1976

Author : Barbara Stankowski

Approved by : Gary M. Ratt
Thesis Advisor

Second Reader

Chairman, Department of Computer Science

Academic Dean

ABSTRACT

This thesis describes the design and implementation of a high level, general purpose, interactive graphics subroutine library for the Vector General Interactive Graphics Display System. The lower interface levels of the system and its effects on the development of this high level interface are discussed. The problems and various approaches associated with the development of a general purpose, high level applications subroutine library that is user oriented is outlined. The basic design goals, solutions and recommendations for further expansion of the system are presented. This graphics subroutine library is implemented within the conventions of the C-Programming language and the UNIX operating system as implemented on the PDP-11/50 in the Naval Postgraduate School Computer Laboratory.

CONTENTS

I. INTRODUCTION..... 7

II. DESCRIPTION OF THE VECTOR GENERAL..... 9

A. HARDWARE FEATURES..... 9

III. EXISTING INTERFACE LEVELS..... 11

A. UNIX AND PDP-11/50 INTERFACE..... 11

B. LOWER LEVEL USER INTERFACE..... 13

1. Picture Structure..... 13

2. Display List Structure..... 15

IV. GENERAL INTERFACE DESIGN CONCEPTS..... 16

A. GRAPHICS LANGUAGE DESIGN..... 16

B. CONSIDERATIONS FOR A USER ORIENTED DESIGN..... 19

1. Graphics Primitives..... 19

2. Default Parameters..... 19

3. Error Diagnostics..... 20

V. VECTOR GENERAL USER INTERFACE DESIGN 21

A. PICTURE STRUCTURE AND USER ASSOCIATION..... 21

B. SELECTING NON-LIMITING PRIMITIVES..... 23

C. NAMING CONVENTION..... 24

D. ERROR DIAGNOSTICS..... 25

E. HIGH LEVEL USER ROUTINES..... 25

VI. RECOMMENDATIONS..... 27

VII. CONCLUSION..... 28

APPENDIX A : User Manual..... 29

APPENDIX B : User Interface Routine Descriptions..... 67

BIBLIOGRAPHY.....122

INITIAL DISTRIBUTION LIST.....124

I. INTRODUCTION

This thesis discusses the design and implementation of a high level, general purpose graphics subroutine library for an interactive graphics display system. The subroutine library was designed to support the Vector General Interactive Display System (Vector General) [1], as installed at the Naval Postgraduate School Computer Laboratory.

The Vector General represents a highly sophisticated graphics display terminal with hardware implemented three dimensional rotation, translation and scaling. An alphanumeric keyboard, lighted function switches, control dials and light pen provide the interactive tools of the system. The Vector General is interfaced with a PDP-11/50 computer and is supported by the UNIX operating system.

The actual design and implementation of this high level, general purpose interface is discussed as well as the goals and problems encountered during it's development. The problems included providing a user with the ability to describe and name picture segments, designing functions that would not limit the capabilities of the machine, and how to effectively utilize the existing interface levels. The main goal was the development and implementation of a simple, easy to use, general purpose graphics subroutine library.

A user's manual was written, describing the implemented library functions, so a user can easily utilize the graphic capabilities of the Vector General from a program written in the high level language, C [2]. This is included as Appendix A.

II. DESCRIPTION OF THE VECTOR GENERAL

The Vector General Graphics Display System is an interactive graphic cathode ray tube (CRT) display that is interfaced with a PDP-11/50 computer. The display interacts with a user by displaying pictorial data, programmaticly described by the user, on the surface of the CRT. The system provides both hardware features and external control devices that can be utilized by a user to alter and manipulate the pictorial data being displayed.

A. HARDWARE FEATURES

The cathode ray tube (CRT) is the most widely used graphics display device and the one capable of generating and dynamically changing graphical data. The Vector General display consists of a CRT and has many supporting hardware features [1,3]. The hardware features provide, in addition to a vector generator, a circle-arc generator and a character generator. The system also has hardware implemented three dimensional rotation, translation and scaling. The features are controlled and coordinated by the Vector General display controller. The controller is also responsible for handling communications with the external control devices. These external devices include an alphanumeric keyboard, thirty-two lighted function switches, ten control

dials, and a light pen. Figure 1 is a block diagram of the Vector General Display System.

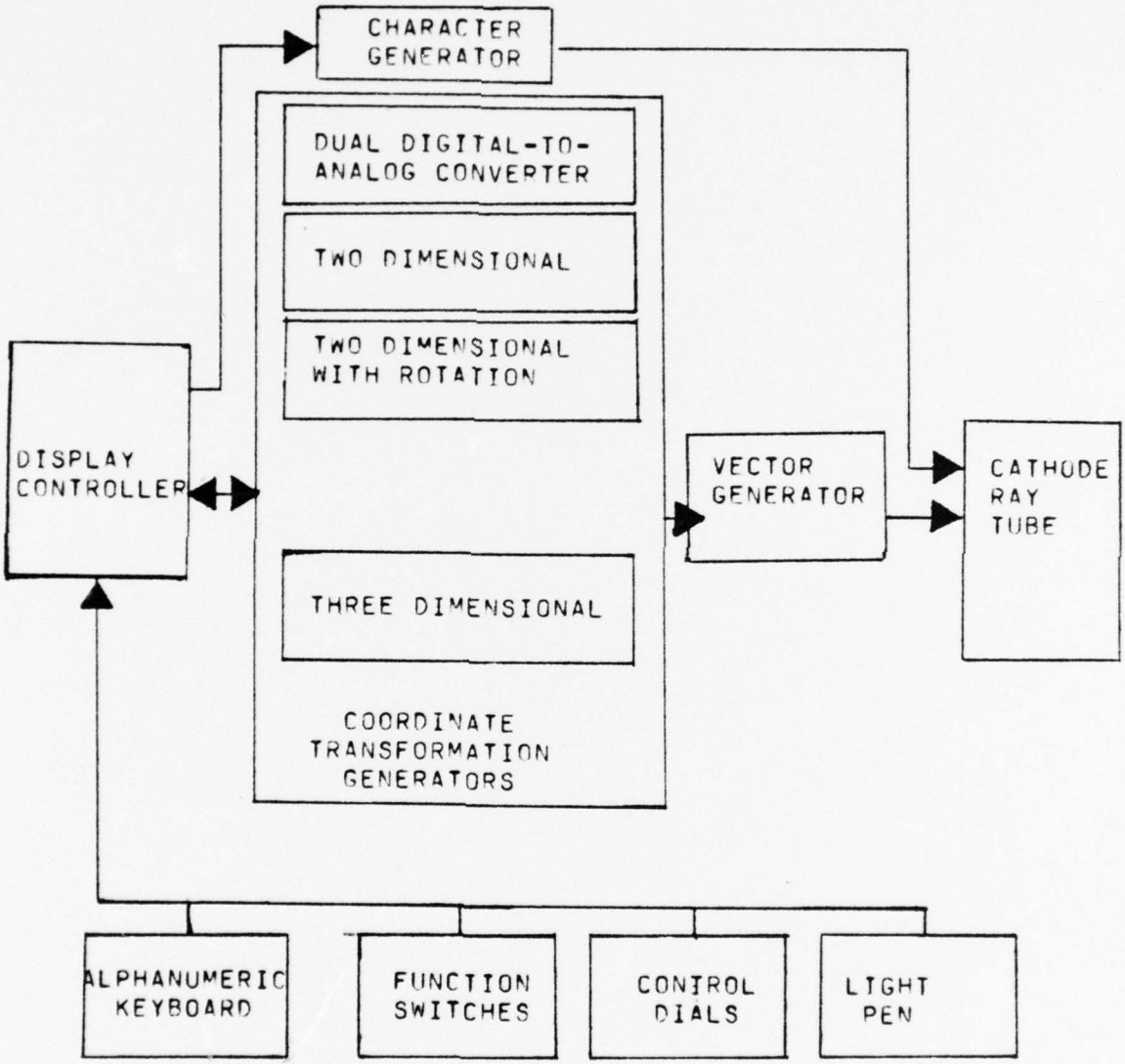


FIGURE 1

Vector General Display System

III. EXISTING INTERFACE LEVELS

The existing interface levels for the Vector General provide an efficient interface with the PDP-11/50 and a user software interface package makes the detailed operations of the system transparent to the user.

A. UNIX AND PDP-11/50 INTERFACE

The Vector General is interfaced with a PDP-11/50 computer having 64K bytes of memory and up to sixteen million bytes of disk storage. The interface was designed to be used in the mutliprogramming environment of the UNIX time sharing operating system [4].

This interface provides the Vector General display processor with access to a 32K block of PDP-11/50 memory. The Vector General display data is maintained within this block, and is continually accessed through a Direct Memory Access (DMA) channel by the Vector General processor in order to refresh the display screen. This approach frees the PDP-11/50 processor for other tasks, for example executing user programs. A detailed description of this interface design can be found in the Design Manual for the Vector General Display Unit [5].

Perhaps the most important aspect of this interface is the resulting support of the Vector General by the UNIX operating system. This added dimension provides the graphics user with all the features of a general-purpose, multi-user, interactive operating system in addition to the graphic capabilities provided by the Vector General.

The UNIX operating system is designed to provide a user with a system that is simple and easy to use [4]. Within this framework extremely powerful features are available to the user. These include: a hierarchical file system, a system command language, compatible file-device and inter-process I/O, over 100 subsystems including several languages and the ability to initiate asynchronous processes. The system's primary high level languages are C and Fortran.

These features of the UNIX operating system provide an efficient and flexible environment for a graphics user. The user has the facilities to maintain and manipulate large graphical data files. Communications links with devices such as tape drives and disks are easily established. The command language provides for the creation, reading and writing of files and for transferring them between devices. An interactive text editor is available for program creation, as is an interactive debugger for program debugging. These are only a few of the extra facilities that become part of the graphic users environment due to this interface with UNIX and the PDP-11/50.

Moving the Vector General from a self supporting stand alone graphics environment, to one supported by a powerful operating system, greatly extends the resources available to the user. This extended system provides an environment for the development of a truely general purpose, high level graphics display system.

B. LOWER LEVEL USER INTERFACE

The existing user interface software package defines high level constructs which the interface routines convert to Vector General commands. These constructs define a hierarchical picture structure within which a user can identify meaningful picture segments. The three construct levels defined by the softeware interface are: picture, object and element [6].

1. Picture Structure

The construct level, picture, refers to all the data that is to be displayed on the Vector General display screen. An element refers to the smallest picture entity that can be independently referenced and changed without affecting the remainder of the picture. Each element is defined by a series of Vector General display instructions. An object represents the lowest construct that can be displayed alone. Each object consists of one or more user defined elements. An object is independently rotatable, translatable and scalable to any portion of the display

screen. The picture defines the picture scale and screen coordinates for all objects.

The generation and contents of the construct, picture and object are the responsibility of the interface software. A series of user routines are provided for the programmer to establish the desired object, element associations. Additional routines are available to dynamically modify picture, object and element parameters.

These three constructs provide a logical base for structuring of pictorial data by a user. It does not, however, produce an optimal user interface. The present interface provides no assistance in the creation of the display instructions that describe each element. The user is responsible for correctly dimensioning an array for each element and for filling it with the correct Vector General display instructions. These instructions require the user to specify coordinate values with respect to the screen coordinate system, and to specify coordinate registers and the desired action(i.e., load, draw or move). This process of describing elements does not provide a user with a clear, simple and unified set of concepts for developing graphic displays. It requires the user to concentrate on the graphics device, its registers, coordinate system and capabilities and not on the picture that is to be created. The process is both tedious and error-prone. The existing interface does not provide an optimal user enviornment, although it does offer a base for a higher level user interface.

2. Display List Structure

The hierarchical picture structure provided by the three construct levels is supported by the implementation of a segmented display list [5]. This allows for the dynamic modification of each construct level which is a necessity for an interactive graphics system [7]. A significant aspect of this implementation is that it supports the concept of shared display code. This is analagous to a conventional subroutine call which allows sections of code to be used repeatedly. Thus, a user can define one element structure and by associating it with an object several times, cause the element to appear repeatedly in a picture.

IV. GENERAL INTERFACE DESIGN CONCEPTS

The existing user interface levels supporting the Vector General and the environment in which it resides has been briefly outlined. The present system does not provide an adequate interface, but does provide the base for the further development of the system. The environment created by the UNIX operating system also supports the concept of a high level graphics interface for the Vector General display. The tools for constructing a high level interface were available, the decision now was how to design an interface package that could utilize the existing environment and provide an optimal user interface.

A. GRAPHICS LANGUAGE DESIGN

In designing a high level interface the main consideration is the development of a high level graphics programming language. The need for such a language for the construction and manipulation of graphical data cannot be overlooked. The ease with which a system can be programmed is a major factor in determining how a system will be utilized [8].

A graphics language must provide a method for describing pictures [9]. The language must have facilities for describing not only non-geometric entities in a simple form but also provide a way to describe two and three dimensional

geometric entities. Additional facilities should be available for scaling, translation and rotation [10].

Developement of a high level language that provides these features can be approached in several different ways. A set of graphic functions or subroutines can be developed to be used by an applications program, written in a high level programming language. Another approach is to utilize the high level language of the host computer and extend it to perform a variety of graphic functions. This extension can be accomplished by changing the existing compiler to handle graphics functions or by developing a preprocessor. If a suitable high level language is not available on a system, then the development of a language specifically for graphics could be considered. This language would require not only the development of graphic functions but must also provide algorithmic-type statements, procedures or subroutine capabilities, and should be interactive [9].

The goal for the Vector General was the development of a high level general purpose language. The UNIX operating system provided the high level language C, that has facilities for handling many different data types, adequate control structures and data structures for algorithm representation. It also provides a simple subroutine calling format. Because of the availability of the high level language the development of a new graphics language was rejected. The decision was to either incorporate graphical functions into this host language by extending the language or by

developing a subroutine library package. In general, several reasons can be stated for not selecting the subroutine library approach. These include the lack of convenience of using subroutine calls exclusively, inefficient data structures provided by the host language, and lack of facilities in the host language to support a wide variety of applications [11]. The C-language provided all the necessary facilities and convenience. The simplest, and most logical approach was the implementation of a high level subroutine package utilized by an applications program written in C.

The process taken in developing this subroutine library is outlined by W. M. Newman and R. F. Sproull [12] :

1. Select a suitable language on which to base the system.
2. Design a set of functions for graphical input and output.
3. Write a programmer's manual.
4. Write the software, to perform the graphic functions.

This approach was taken to insure the development of a user oriented graphics design rather than one that was merely easy to program and implement.

B. CONSIDERATIONS FOR A USER ORIENTED DESIGN

One of the goals of this interactive subroutine library is to provide a general purpose high level package that will support a wide variety of applications. Additionally, it is important that the system be user oriented. This means that the resulting graphics programs should be as easy to write and maintain as any other interactive program. The design has to provide a clear and vivid means of describing the pictures a user wants to create [13].

In trying to produce this type of user oriented package several requirements have to be met by the design specifications.

1. Graphics Primitives

A graphics system should provide a small number of powerful graphics primitives [8]. These should be designed in such a way that the user is required to concentrate on the picture being described rather than on the hardware features of the machine. The user must be able to construct logical picture segments, combine them into a meaningful picture and then manipulate any combination of these picture segments [13].

2. Default Parameters

Default parameters should be skillfully introduced into the system [8]. This eliminates a novice user from the irritating details of the system. If default values are

automatically included, a novice user can concentrate on the basic problem of describing a picture.

3. Error Diagnostics

User paramaters should be checked and all errors should provide informative diagnostics. Errors should only activate diagnostic routines, and they should never force the termination of a user program [10]. A user should be able to trace and correct errors with a minimum of effort.

V. VECTOR GENERAL USER INTERFACE DESIGN

The design of the Vector General high level interface is implemented in the form of subroutine library functions which are utilized through programs written in the high level language C. Every effort was made to limit the number of functions without limiting the hardware capabilities of the Vector General. The design of this primitive set is based on the concept of picture structure, as defined by the constructs picture, object and element.

A. PICTURE STRUCTURE AND USER ASSOCIATION

The present interface provides the capability of describing pictures as a collection of objects with associated elements and attributes. The framework for this logical description of a picture and the implementation of a structured display file is provided by the existing interface. This is, however, inadequate because within this framework a user cannot easily generate the actual display instructions that make up each element. Additional capabilities for easily describing a logical picture element and its association with other elements and with the picture is provided by the design described here.

The implemented design introduces the concept of an element block. An element block consists of a series of move-draw instructions or of ASCII character data. This approach provides a flexible way for clearly isolating and naming specific picture segments. Additionally the responsibility of dimensioning arrays for each element display list is removed from the user and is instead managed by the software interface. This dynamic allocation and management of display lists by the software interface greatly extends the flexibility of an element block. Each element block can contain not only graphic functions but any of the C-language arithmetic, conditional or logical statements, as well as subroutine calls. A user can also define a graphical entity within an element block recursively.

Many problem solutions are most appropriately defined in recursive terms. Wirth sites several examples of graphic patterns that are most easily described by a recursive algorithm [15]. The capability of defining elements by recursive algorithms provides the user with another powerful tool.

In addition to the element block structuring, a primitive is provided so that the user can easily establish the desired element, object relationships. This provides an easy and concise method for logically constructing a picture. With this primitive any number of element blocks can be associated with an object at one time or at several different times within a program.

B. SELECTING NON-LIMITING PRIMITIVES

The Vector General provides 12 different vector types, arcs, circles, ASCII characters and has the facilites for creating three dimensional images. Incorporating the flexibility of 12 vector types and two or three dimensions within a limited set of graphic functions presents several problems. The hardware capabilities of the Vector General must be available to a high level user in a concise and logical manner. Extending the function set to include separate functions for each vector type and for two or three dimensions, is not appealing because it provides a user with too many opportunities for meaningless and erroneous operations [12]. The other possibility is to use variable length parameter lists for a small function set. The C-language supports the use of varible length parameter lists and this approach is implemented without compromising other factors such as ease of use, simplicity, and understanding. This approach lead to the development of nine functions which, in combination, allow utilization of all of the Vector General facilities in creating an element block and establish the desired object, element linking. A complete description of these functions can be found in Appendix A.

C. NAMING CONVENTION

The constructs object, element and picture provide a multi-level naming structure [14]. This structure allows a user to associate integer names with each object and element that is created. These integer names are assigned values by the user interface. Additionally each element is also associated with an array name. This naming convention, which requires user-interface interaction is inadequate for a high level interface. A naming convention that allows a user to independently assign meaningful names to objects and elements is required. The high level interface design described here provides a simple naming process which requires a user to specify an element or object name as a quoted character string within a parameter list. A name is associated with each element and object as they are created by the user. There are no limitations imposed on the user in assigning names except that each name must be unique. Each object and element name is used throughout the program to reference a specific picture segment. This implementation provides a flexible and easy method for a user to associate meaningful names with each object and element.

The existing interface supports the concept of shared display code, thus allowing for the repetition of an element in a picture. This repetition is accomplished by associating an element with several objects or with one object several times. This latter case requires that a user be able to uniquely reference each occurrence of an element

within a specific object. The naming convention had to be extended to handle this situation. A primitive has been designed that allows a user to assign any number of unique aliases to a specific element block. An element can be associated with an object several times, each association, however, is established by using an alias. In this way each occurrence of an element within a specific object can be uniquely referenced by its alias.

D. ERROR DIAGNOSTICS

All functions and user routines provide error checking and diagnostic information. An error will never cause the termination of the user's program. Every effort is made to allow a user's program to run to completion, so a user will receive some visual feedback from the Vector General display screen. A picture, even one that results from several errors can act as a useful debugging aid in itself.

E. HIGH LEVEL USER ROUTINES

In order to provide a comprehensive graphics package, the naming convention and picture structure concept has to be applied to the lower level user routines that dynamically change and manipulate picture segments. These routines are incorporated into this high level structure. When possible several low level routines are combined to provide the user with a simpler and yet more powerful routine. For example, separate routines for the deletion of an object and element

from the display screen are provided at the lower level. These two routines are combined into one routine that erases the entire picture, an object, or any number of elements associated with a specific object. Additionally, a routine is included so a user can not only erase an element block but also free up the memory locations associated with the element block.

The advantages of allowing a user to manipulate display list storage lies in the fact that it is extremely difficult for the software interface to determine which display list structures are no longer required and should be removed. The user on the other hand knows precisely when a specific item is no longer needed [16]. The system makes available a finite area for the creation of display lists. Erasing an element does not release memory. Elements still exist and can be reassociated with the picture at anytime. The user should have the ability to optimize this storage area, by releasing elements that are no longer required. This is particularly important when a user program exceeds the allocated display list storage. Only by selectively removing elements from memory and releasing the related memory locations can new elements be created. A user can extend the systems storage limitations by efficiently managing the display list storage area.

VI. RECOMMENDATIONS

The implemented graphics subroutine library provides a general purpose interactive graphics package. The routines provided for using the external display control devices are adequate, but can be extended to provide a simpler package for a user. For example, the routines that control the function switches and the function switch lamps could be combined to provide a simpler interface. A variety of routines for utilizing the light pen should be available to a user. For example, a light pen tracking routine would benifit the system.

VII. CONCLUSION

The high level general purpose graphics subroutine library discussed in this thesis is operational. Initial test programs indicate that the system does provide a simple, but powerful approach for the development of interactive graphics programs.

APPENDIX A: USER REFERENCE MANUAL

I. INTRODUCTION

The vector general is an interactive graphics display system which has been interfaced with the PDP-11/50 computer. The display interacts with a PDP-11 user by displaying pictorial information on the surface of a cathode ray tube and by accepting information from its external control devices. The external devices consist of an alphanumeric keyboard, 32 lighted function switches, 10 control dials, and a light pen. Through a C-callable interactive graphics program library, the pictorial information desired by a user can be described, altered and manipulated. This manual will not discuss in detail the electronic functions of the vector general, or the vector general's interface with the PDP-11. The purpose of this manual is to instruct a user in the creation and manipulation of pictorial data on the vector general display.

This manual describes the use of a C-callable interactive graphics program library. A knowledge of the C-programming language is assumed. The user is also directed to Appendix B for a brief description, calling format and error diagonistic information on each of the user interface routines.

II. THE VECTOR GENERAL DISPLAY SYSTEM

A more detailed discussion of the vector general can be found in the Users Manual for the Vector General Display Unit [6] and the Design Manual for the Vector General Display Unit [5].

A. THE DISPLAY

The vector general is a cathode ray tube (CRT) display on which a visible pattern can be created by the movement of an electron beam. The electron beam causes a florescent spot to appear on the face of the display tube. The movement of the beam is controlled by a method called random scan, which in effect steers the spot in a straight line between two points on the display screen. The resulting line or vector, combined with others, creates a picture or pattern on the display screen.

To maintain a clear picture on the display screen requires that the pattern be redrawn on the tube repeatedly at approximately thirty to forty times a second. Each repetition is called a frame and the frequency at which it is redrawn is called the refresh rate. If the pattern is not repeated often enough, or more information than can be redrawn in a frame is displayed, a distortion of the picture will occur on the display screen. This distortion is called flicker.

B. HARDWARE FEATURES

The system has several hardware features, in addition to a vector generator, which greatly extends its capabilites. These include, the ability to produce three dimensional figures, an ASCII character set, and the hardware generation of arcs and circles. Other features provide the hardware mechanisms for the rotation and translation of user specified picture segments. These hardware features are controlled and coordinated by the display controller. The controller is responsible for handling the communications between the user interface, the external control devices and the display hardware.

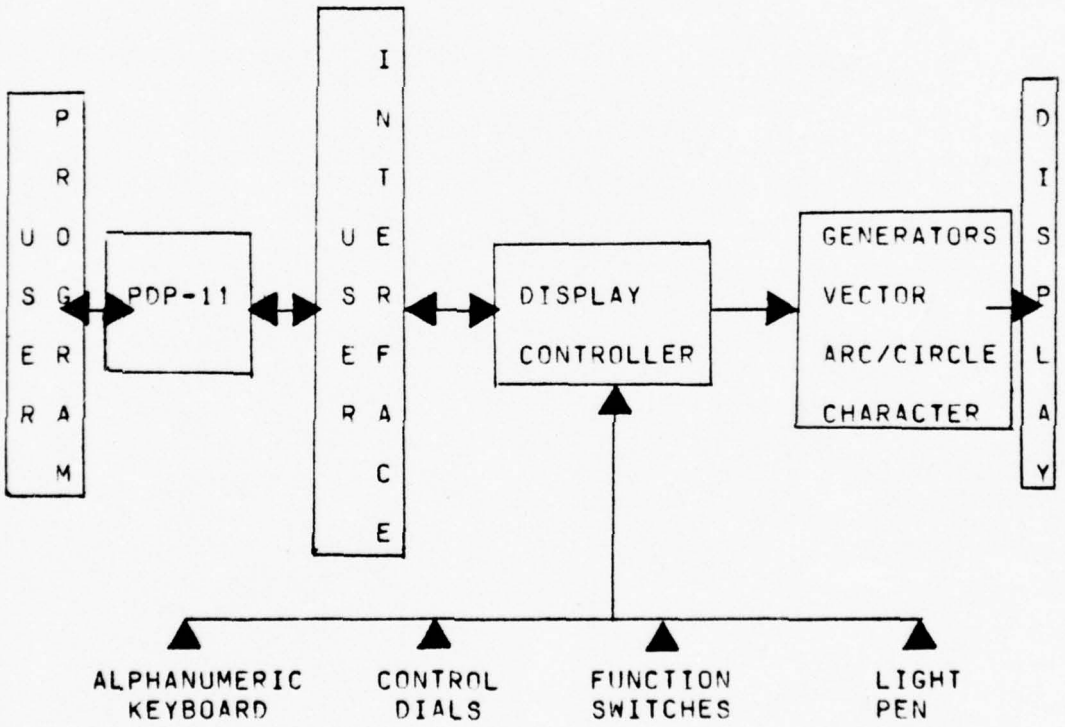
The main purpose of the external control devices is to facilitate user interaction with the display. These devices include an alphanumeric keyboard, 32 lighted function switches, 10 control dials and a light pen.

Additional information on the vector general hardware can be found in the Graphics Display Reference Manual [1] and the Graphics Display System Technical Manual [3].

C. THE SYSTEM INTERFACE

A vector general user defines pictorial data and its manipulation within a C-language program. The execution of the program, on the PDP-11/50, causes the vector general software interface to be activated. It is this software

interface package, which communicates user requests and receives information from the vector general controller. The interface package passes user requests, in the proper form, to the display controller. The controller will activate the proper generator (ie. character, vector, arc/circle) which will output the desired information, requested by the user, on the display screen. The controller will also pass information from the external control devices back to the user via the interface software. This relationship is illustrated in figure A-1.



Interface Relationship with the Vector General
FIGURE A-1

D. INTERFACE WITH EXTERNAL DEVICES

Each of the the external interactive devices communicate directly to the vector general controller. Information from these devices is returned to the user via the controller and the user interface. The user program may utilize the information returned from these devices to control program flow. This allows a user to interactively control and manipulate the pictorial information at display time. Specific user routines which activate these external devices and provide a communications channel with the user are discussed in detail in later sections of this manual.

1. Alphanumeric Keyboard

The alphanumeric keyboard allows the user to input information in the form of ASCII character codes. Through the user interface the user can display the information on the vector general display screen. The data entered from the keyboard can also be returned to the user program for processing.

2. Function Switches

The 32 lighted function switches provide the user with information which can be used to interactively manipulate pictorial data at display time. Each function switch can be assigned specific meaning by the user program. The user interface returns, from the controller to the user, information on which function switches have been pressed. A user program could use this information to selectively

rotate, translate or perhaps scale particular picture segments.

3. Control Dials

The 10 control dials provide numeric information to the display controller, specifying the degree to which each dial has been turned. This information, through the user interface, can be provided to the user. A user program may utilize the values of the variable control dials in determining the distance or rate at which a portion of the picture may be moved or rotated.

4. Light Pen

The light pen, a wand containing a photo cell, can be used to selectively point to different picture segments on the display screen. The interface provides a user program with information on which picture segment was pointed to by the light pen. A user program can turn the light pen selectability of specific picture segments on or off. For example, an interactive user might select sections of a picture for erasure by pointing to them with the light pen.

III. INITIALIZATION

A. INTERFACE INITIALIZATION

The vector general display system and the interface software with the PDP-11 must be initialized before any data can be displayed. The initialization routine sysinit must

be called before any other routines are utilized. This routine sets all the system default parameters, such as the screen coordinate system.

If for some reason the initialization cannot be completed the user program will be terminated. This error usually occurs because another user is accessing the vector general.

8. DISPLAY INITIALIZATION

1. Coordinate System

The user can specify a two or three dimensional cartesian coordinate system, of any scale. All display coordinate values referenced by the user will be interpreted according to this coordinate system definition. A user may redefine the coordinate scale at any time in a program. The user will define the coordinate system in a call to the routine coordsys.

```
coordsys(dim,minx,maxx,miny,maxy [,minz,maxz]);
```

The routine requires the user to specify if the coordinate system is to be two or three dimensional and the range of each coordinate. If the parameter dim is two, indicating a two dimensional coordinate system is desired, the range of the z coordinate can be omitted, and will be ignored if it should be included.

If this routine is not called by the user the default coordinate system will be used. This default system is

defined as three dimensional with the x, y, z coordinates ranging from -100.0 to 100.0. All coordinate values will be interpreted by this default system when coordsys is not called by the user.

2. Picture Scale

The rectangular, 13 by 14 inch, portion of the display screen that can be viewed by the user is called the visible space. The maximum picture space is larger than the visible space, covering an area of 30 by 30 inches. This extra area allows a user to rotate or move part of the picture to the extreme boundries of the visible space without any distortion. It also permits limited zooming.

The pictorial data being displayed can be adjusted in size, or scaled, by two different controls. One, the gain control dials on the vector general display unit allow the user to manually manipulate the picture scale. The second provides scale control within the user's program by calling the routine vapscal. This routine is discussed in detail in Section V of this manual.

IV. CREATING A PICTURE

A. PICTURE STRUCTURE

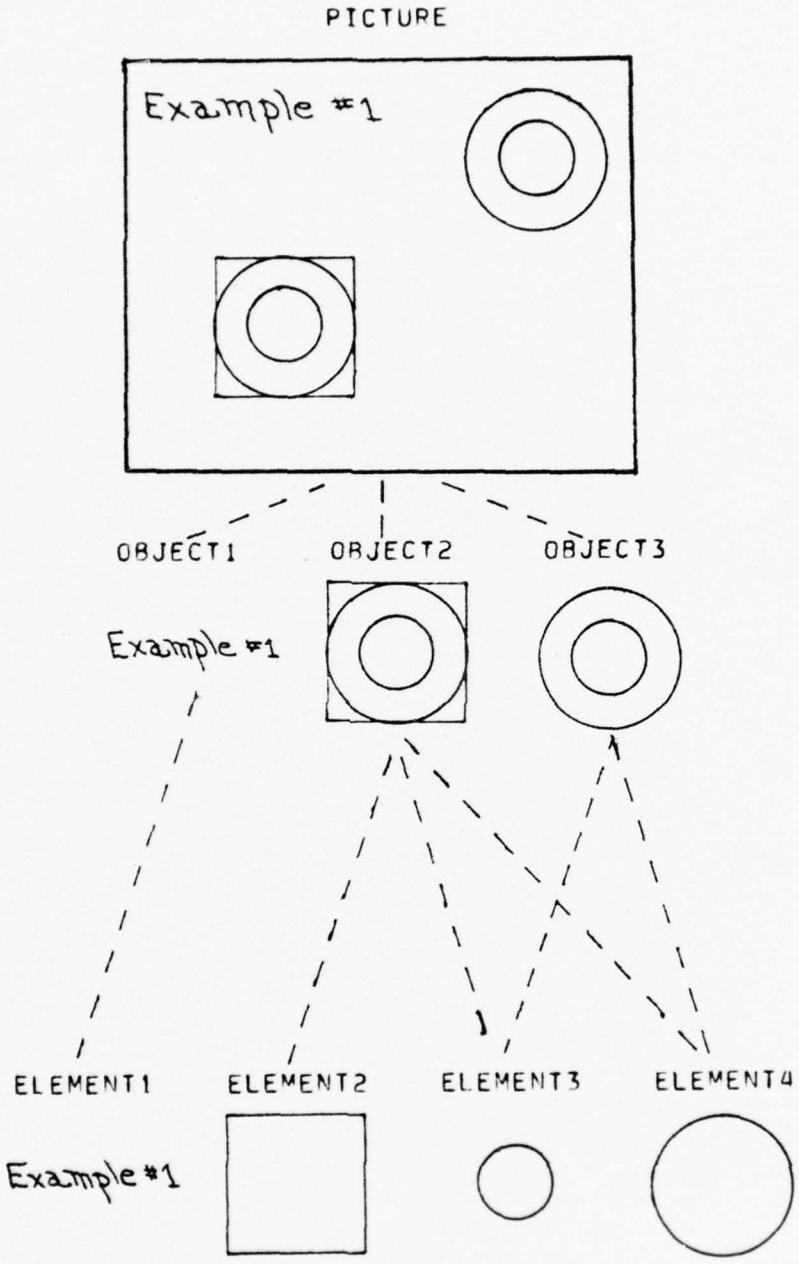
All of the information that a user desires to display on the vector general must be incorporated into the

hierarchical picture structure defined by the user interface package. The three hierarchical levels are defined as: picture, object, element. These levels specify the underlying structure of the graphical display and determine the operations a user can perform on information associated with each level.

The term picture refers to all of the data that is to be displayed on the vector general display space. The term element refers to the smallest picture segment which can be independently referenced and changed without affecting the remainder of the picture. Each element, or independent picture segment, has a unique name associated with it. A collection, or meaningfull grouping, of elements is called an object. Each object is also labeled by a unique name so it can be easily referenced by a user. Figure A-2 illustrates the relationship of the three levels in the actual structure of a picture.

An element is completly described and named by the user. It can describe either graphical or ASCII character data. An element can be independently added or erased from the display screen. It can be caused to blink or be specified as being light pen selectable.

Each element must be associated with at least one object before it can be displayed on the vector general. An object can consist of one or more elements. A user is reponsible for establishing the desired object-element association,



Hierarchical Picture Structure

FIGURE A-2

and for specifying a unique name for each object. Each user defined object can be independently rotated, or translated to any section of the 30 by 30 inch display space. An object can be added or erased from a picture, scaled, and specified to blink or to be light pen selectable. Each object's intensity can be varied in order to give three dimensional objects their depth queuing. These actions, when applied to a specific object, affect every element that has been associated with it by the user.

A picture can contain one or more objects. The coordinate scale and picture scale defined by the user affects the entire picture. A picture's coordinate scale can be varied but this action will affect every object defined as part of the picture. An entire picture can also be erased, specified to blink or be light pen selectable.

A summary of the operations for each level of the hierarchy is outlined in Figure A-3.

B. CONSTRUCTING AN ELEMENT

Every element is completely described by the user within an element block. There are two types of element blocks. A draw element block, describes graphical information. The other, a character element, describes ASCII characters that are to be displayed. Each element is uniquely named and this name will be used to reference this particular structure.

PICTURE:

- Define picture coordinate system
- Picture scale
- Erase
- Remove
- Blink
- Light pen selectable

OBJECT:

- Translate
- Rotate
- Scale
- Intensity scale
- Intensity offset
- Erase
- Remove
- Blink
- Light pen selectable

ELEMENT:

- Erase
- Remove
- Blink
- Light pen selectable

Summary Operations Associated with Each Hierarchial Level
FIGURE A-3

1. Draw Element Block

A draw element block represents a group of draw instructions that describe a specific structure, or picture segment. These draw instructions include setvector, move, line, arc and circle. A draw element block begins with a call to the routine drawele and is terminated by a call to the routine endele. The draw instructions that are executed between drawele and endele describe the actual picture segment.

The only parameter required by drawele is a quoted character string, or pointer to a character string, specifying the name the user wants to associate with this element. This name will be used throughout the program to reference this element block.

The basic draw element block, and the related draw instructions are represented in the following format:

```
-- drawele("element-name");
|       setvector(vtype,vmode,[inc],[scale]);
|       .
|       .
|       .
|   ---setvector(vtype,vmode,[inc],[scale]);
|   |   move(x,[y],[z]);
|   |   line(x,[y],[z]);
|   |   circle(dir,centx,[centy],[centz]);
|   --- arc(dir,centx,[centy],[centz],endx,[endy],[endz]);
|       .
|       .
|       .
--- endele();
```

The user can select one or more of the twelve vector

types in constructing an element. These vector types describe how the coordinate data will be interpreted in drawing a vector on the display screen. The choice of a vector effects the parameters that will be passed in each of the move, line, circle or arc instructions. The user specifies a vector selection by calling the routine setvector.

a. setvector - This draw instruction must be called immediately after drawele, and may be called any number of times within the element block. Each setvector, and the draw instructions that follow it, comprise a subgroup. The setvector instruction determines the manner in which the line, move, arc and circle instructions in the subgroup will be interpreted, as well as their visual appearance on the display screen. The routine is called with the following parameters:

```
setvector(vtype,vmode [,inc] [,scale]);
```

The parameter vtype specifies which one of the twelve vector types the user wants the following group of line, move, arc and circle instructions to utilize. The parameter vmode indicates the vector mode, or appearance of the vectors to be drawn (ie, solid line, dotted line, etc.). Certain vector types require additional information, this information is specified by the parameters inc and scale.

TABLE A-I

SUMARY OF VECTOR TYPES

NAME	DESCRIPTION
VA	- vector absolute, each coordinate is specified with respect to the origin. Each point(x,y,z) references a unique point on the display screen.
VAX	- vector absolute auto-increment x, every draw instruction causes the y and z absolute values to be updated while x is stepped by a constant value.
VAY	- vector absolute auto-increment y, every draw instruction causes the x and z absolute values to be updated while y is stepped by a constant value.
VAZ	- vector absolute auto-increment z, every draw instruction causes the x and y absolute values to be updated while z is stepped by a constant value.
VR	- vector relative, each x, y, z coordinate value indicates the amount that is to be added or subtracted from the previous absolute coordinate point.
VRX	- vector relative auto-increment x, each draw instruction causes the y and z coordinate values to be incremented by the specified value while x is stepped by a constant value.
VRY	- vector relative auto-increment y, each draw instruction causes the x and z coordinate values to be incremented by the specified value while y is stepped by a constant value.
VRZ	- vector relative auto-increment z, each draw instruction causes the x and y coordinate values to be incremented by the specified value while z is stepped by a constant value.

TABLE A-I
(continued)

INC2- two dimensional incremental vector, a relative vector that optimizes storage requirements. The coordinate increment values are limited to values approximately 3% of the user's coordinate range.

INCX- two dimensional incremental auto-increment x, is a relative vector that optimizes storage requirements. The y coordinate value is incremented by a small value while x is stepped by a constant value.

INCY- two dimensional incremental auto-increment y, is a relative vector that optimizes storage requirements. The x coordinate value is incremented by a small value while y is stepped by a constant value.

INC3- three dimensional incremental vector, a relative vector that optimizes storage requirements. The x, y, z coordinate increment values are limited to values that are approximately 3% of the user's coordinate range.

A brief summary of the twelve vector types is listed in Table A-I, a more detailed account of each vector type and the parameters required by setvector are listed in Appendix B. Figure A-4 illustrates the five different vector modes that are available.

VECTOR MODE	PARAMETER VALUE	VISUAL APPERANCE
line	LN / 00	———————
dashed	DSH / 020	— — — — — —
dotted line	DOT / 040	- - - - - - - - -
end point	PNT / 060	.
dash-dot-dash	DD / 0120	— - — - — - — -
dash-dot-dash	DDD / 0140	— — — - — - —

Vector Modes

FIGURE A-4

b. move - The draw instruction move is used to reposition the beam on the display screen. It will produce no visible line or pattern. The format of the instruction is:

move(x [,y] [,z]);

The coordinate values x, y, z will be either absolute or

relative values. The vector type selected in the proceeding setvector instruction will determine how the value of these parameters will be interpreted. The bracketed values indicate parameters that may be optional.

If the user coordinate system is not three dimensional, the z parameter can be omitted, and will be ignored if it should be included.

c. line - The line instruction, draws a visible line or vector on the display screen. The line is drawn from the present beam locaton to the specified end point. The format of the instruction is:

```
line(x [,y] [,z]);
```

The coordinate values x, y, z will be either absolute or relative values. The vector type selected in the previous setvector determines how the value of these parameters will be interpreted.

If the user has defined a two dimensional coordinate system, the z parameter can be omitted, and will be ignored if it should be included.

d. circle - The circle instruction will draw a circle beginning at the present beam location about the center point specified by the user. The difference between the present beam location and the center point determines the radius of the circle. The instruction is used by the following format:

```
circle(dir,centx [,centy] [,centz]);
```

The parameter dir indicates in which direction the circle is to be drawn, clockwise or counterclockwise. The number of parameters required and their values are determined by the vector type selected in the previous setvector instruction. A circle cannot be drawn by any of the four incremental vectors.

If the user coordinate system is two dimensional, the z parameter can be omitted, and will be ignored if included.

e. arc - The arc instruction will draw an arc from the present beam location, about the specified center point, to the desired end point. The distance between the starting point and the center point determines the radius of the arc being drawn. The instruction format is :

```
arc(dir,centx [,centy] [,centz],endx [,endy] [,endz]);
```

The parameter dir, gives the direction the arc is to be drawn, clockwise or counterclockwise. The coordinate values are determined by the vector type selected in the previous call to setvector. An arc cannot be drawn by any of the four incremental vectors.

If the user coordinate system has been defined as two dimensional, the z parameters can be omitted, and will be ignored if included.

If the distance from the starting point of the arc to the center point, and the distance from the end point to the

center point are not equal, the resulting arc will contain a straight line. The straight line results from the arc generator trying to compensate for the two different distances to the center point. One distance will be used to determine the radius of the arc, the arc will then be drawn using this radius. The arc will stop at the systems new defined end point and a straight line will be drawn to the end point that had been specified by the user. The resulting arc appears in the following form:

2. Character Element

A character element represents ASCII character data that is to be incorporated into the picture structure. A user can specify a character element containing ASCII, special vector general characters and formatting symbols to be displayed on the vector general display screen. A user can select from four character sizes and has the option of selecting a slanted character set. The text can be displayed horizontally or vertically on the screen. The vertical position causes the characters to appear as if they were on a page that had been rotated ninety degrees counter clockwise. The user can select the position on the screen where the string is to begin, or can output it relative to the present beam

position.

Each character element is given a unique name by the user. This name will allow the user to easily reference each character element. A character element is represented by the following format:

```
charele("element-name",string,size,wdir,slant,x,y);
```

The parameter string can be either a quoted string within the parameter list or a pointer to a character string or array. The character string will begin at the point(x,y) or can be output relative to the present beam location by replacing the x and y parameter with the constant VGREL. For example, the following character element, when linked to an object, would be output relative to the present beam position:

```
charele("element-name","Now is the time",SZ4,HOR,SLNT,VGREL);
```

A summary of the character element parameters is presented in Table A-II. The character set available on the vector general is illustrated in Figure A-5. All of the vector general characters can be represented within a character string. The special formatting symbols and symbols for the special vector general characters are listed in a Appendix B.

TABLE A-II

SUMMARY OF CHARACTER ELEMENT PARAMETERS

size: specify character size

SZ / 00 - use previously defined character size

SZ1 / 0100 - set size to 100 columns by 60 lines

SZ2 / 0120 - set size to 81 columns by 41 lines

SZ3 / 0140 - set size to 60 columns by 30 lines

SZ4 / 0160 - set size to 32 columns by 16 lines

wdir: write direction

HOR / 00 - write characters horizontally

VER / 0200 - write characters vertically

slant: specifies regular or slanted characters

SLNT / 00 - slanted characters

NSLNT/ 01 - regular characters

Character Set

FIGURE A-5

C. LINKING ELEMENTS TO OBJECTS

Draw and character elements represent the smallest picture segment that can be independently referenced by the user. In order to display an element it must be associated with at least one object. An object represents the smallest entity that can be displayed independently on the vector general screen.

1. Object Routine

Each object is given a unique name so it can be easily referenced by the user. Elements can be linked to an object at one time or by several different calls to the routine called object. An object can consist of one or more elements. A specific element can be linked to several different objects, or may be linked to one object several times. This object-element association or linking is established by the routine object. This routine has the following format:

```
object(num,"object-name","element-name",...,"element-name");
```

The parameter num indicates the number of elements that are to be linked to the named object by this call. The elements are referenced by the names specified, by the user, in the preceding drawele and charele routines. Since each object is associated with a unique name, a second call to

object, with a duplicated object name, will cause the elements to be added to the object first associated with that name.

When an element is linked to a specific object several times, the user can no longer reference a specific occurrence of this element within the object, for example:

```
object(3,"Tree","branch","branch","branch");
```

In this case the element branch has been linked to the object tree three times. When displayed, element branch will appear three times. Now ,however, the user cannot uniquely reference a specific occurrence of the element branch. If the element called branch was selected to be erased, the first occurrence of branch would be erased from the screen. In many instances it may desirable to uniquely reference each occurrence of an element within a specific object. This can be accomplished by associating several unique names with an element. The routine copyele provides this capability.

2. Copyele Routine

This routine allows a user to assign several unique names to a specific element structure. In this way, an element structure can be associated with an object several times and each occurrence can be uniquely referenced. The routine is represented by the following format:

```
copyele(num,"element-name","copy1-name","copy2-name",...);
```

The parameter num indicates the number of additional names a user wishes to associate with the named element structure. The "element-name" refers to a previously defined draw element or character element block. Each of the "copy-names" must be unique.

Now reconsider the previous example. If the element branch is associated with two other unique names and these three names are used in the object-element linking, each occurrence of the structure can now be uniquely referenced by the user. The following two statements will accomplish this task.

```
copyele(2,"branch","branch1","branch2");
object(3,"tree","branch","branch1","branch2");
```

V. DISPLAY MANIPULATIONS

A. USER ROUTINES

After the user has incorporated all data that is to be displayed into the desired picture structure, it can now be manipulated and transformed. The following user routines provide the means to manipulate the picture, objects and elements that have been created by the user.

A description of each user routine, calling format and error diagnostics are included in Appendix B.

1. Blink

The display blink mode can be set for the entire picture, single object or for any number of elements associated with a specific object. Modifying the blink mode of an object affects all the elements associated with that object. The routine blink will turn the blink mode on or off for the specified picture segment.

2. Erase

The entire picture, a single object or any number of elements associated with a specific object can be erased from the display screen. The picture segments that are erased from the screen can be redisplayed by again establishing the desired object-element association. This is accomplished by calling the routine object, as described earlier. Erasing an object will affect all elements associated with the named object.

3. Input Data from the Display Keyboard

The routine indata allows a user to receive and output characters from the vector general keyboard onto the display screen. The ASCII character data is also placed in a user specified character array for processing by the user's program. Up to one line of text can be entered, data entry is terminated by a carriage return. The termination of the data entry also erases the output characters from the

display screen.

4. Intensity Offset

The routine intoffset allows the user to vary the intensity level of a three dimensional object, or impose a screen cut-off plane for the named object.

5. Intensity Scale

The routine intscale allows a user to vary the intensity of a three dimensional object, this provides the depth-cueing or shading for a three dimensional object.

6. Light Pen

The light pen selectability of the picture, a single object, or any number of elements associated with a specific object can be turned on or off by the routine called lightpen. This determines what picture segments will be affected by light pen interactions.

7. Remove

The routine remove provides a user with the ability to release the memory locations associated with a specific element structure. The user can remove the picture and release all the memory locations that have been used to describe all the existing elements. Additionally the picture will be erased from the screen. Each element that is removed from memory can no longer be referenced or linked to objects. An individual element can also be removed from memory, this will cause every occurrence of the element to be erased from the screen and the memory locations

associated with the element's description will be released. The user can specify an element for removal by either it's original name or by any of the copy names associated with it. Remove results in the elimination of all occurrences and all copies of an element from the display screen. A element that has been removed can be redisplayed only by reconstructing the element block and by again establishing the desired object-element association.

8. Rotate

This routine allows a user to rotate an object about the x, y, and z axis. The rotation of an object affects all elements associated with the named object. Arcs, circles and characters are always drawn in a plane parallel to the screen, and are rotatable in a three dimensional coordinate system about the z-axis.

9. Scale

The routine scale allows a user to scale independently any object of the picture. All elements associated with the object will be scaled by the specified scale factor.

10. Translate

The routine trans allows the user to move an object anywhere in the display space. The object and all its associated elements can be moved in the x, y, and z plane. Continually translating an object by very small increments will cause it to appear as if it is moving across the display

screen. The following example will move the named object diagonally across the display screen:

```
x = y = .01  //coordinate system is 2D
             //ranging from -1 to 1.
for(i=0;i<100;i++)
{ trans("box", x, y);
  x = x + .01;
  y = y + .01;
}
```

VI. OTHER USER INTERFACE ROUTINES

Additional user routines are available and are described in detail in the Users Manual for the Vector General Display Unit [6]. Only a brief summary of each routine will be presented in this manual.

1. vgclock

This routine allows a user to set the refresh rate of the vector general display. The vector general is automatically initialized with a refresh rate of forty hertz.

2. vgdial

The routine vgdial returns to the user's program the values of the ten variable control dials.

3. vggetcar

The routine vggetcar will return to the user's program a single character entered at the vector general keyboard. If no character has been entered a -1 will be returned. This routine does not display the characters on the vector general display screen.

4. vagetfsw

This routine will return to a user's program the value of the 32 function switches. The values returned will indicate which function switches are depressed.

5. vglamps

This routine is used in conjunction with vgaetfsw. It will turn on the function switch lamp for each function switch that is depressed.

6. vapicture

This routine causes the user's picture to be displayed on the vector general display screen. Nothing will appear on the display screen until this routine is called.

7. vapost

The routine vapost allows the user to transform the pictures coordinate axis. This transformation will cause the entire picture to be repositioned on the display screen.

8. vapscal

The user can modify the scale of the entire picture by calling the routine vapscal.

9. vaterm

This routine terminates the vector general process and releases all the systems resources that were being utilized by the vector general. This routine should be called at the conclusion of all vector general display operations.

VII. RUNNING A VECTOR GENERAL PROGRAM

A. PROGRAM FORMAT

In order to properly utilize the vector general display system, routines to initialize and terminate the system must be called. These routines initialize the vector general, start the actual visual display and properly terminate the display at the end of a users program. Each user program must include a call to these three routines:

```
sysinit();
vgpicture();
vgterm();
```

The first routine, sysinit, must be called by the user before any other user interface routine is called. To display the picture that a user has described on the vector general screen, the routine vgpicture must be called. This routine is called only once, and nothing will appear on the screen until it has been called. Finally, to properly terminate, the system requires a call to the routine vgterm at the end of the user program.

B. TO COMPILE

Once a graphics program for the vector general has been writted in the C-language it must be compiled with the user interface graphics subroutine library. To include this information a user should issue the following command to

properly compile a vector general graphics program:

```
cc -O filename -lv
```

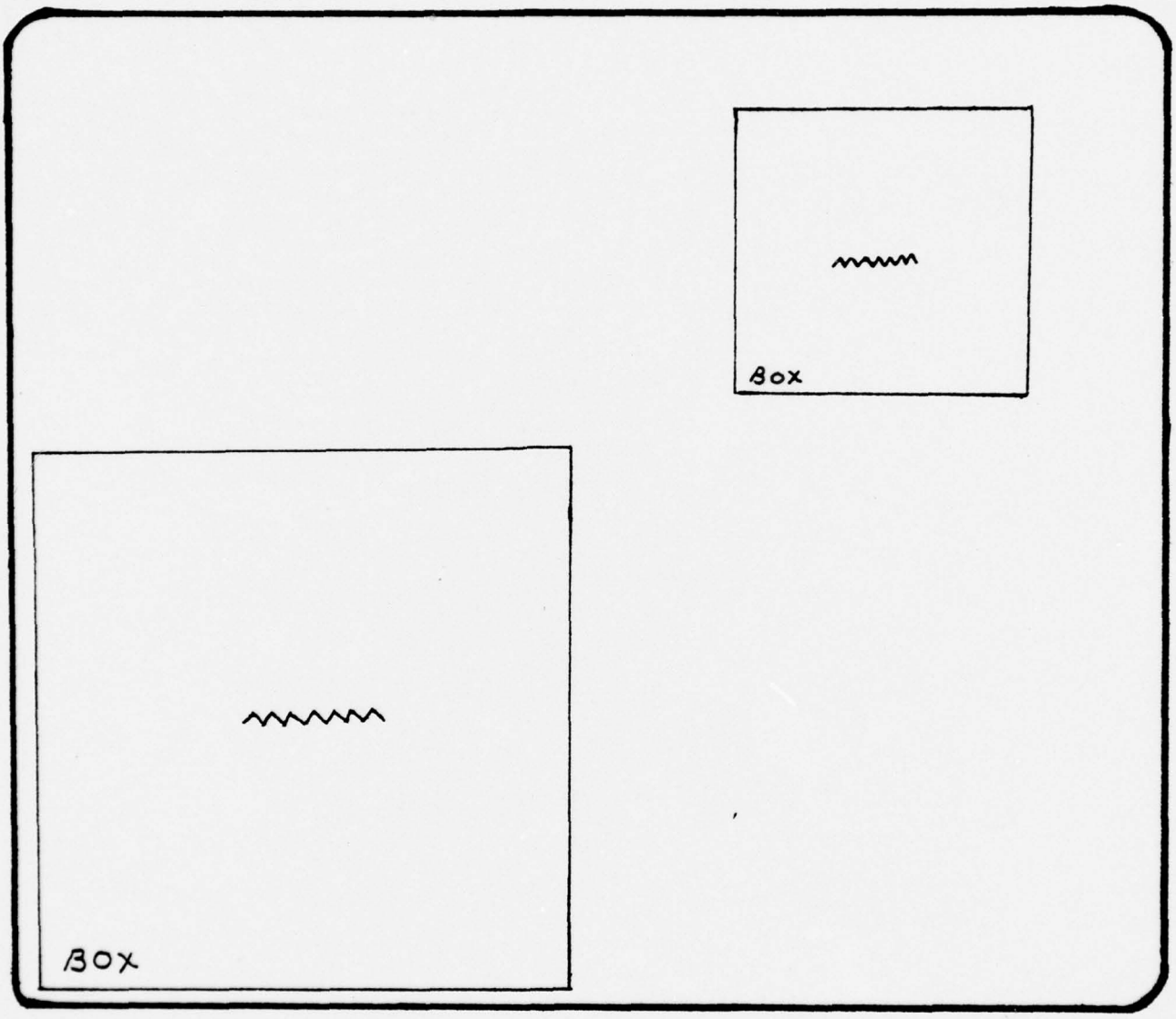
In order to utilize the vector general's program constants used in this manual, the file containing these constants must be included in each graphics program. The following command will include the file:

```
#include "/usr/lib/vgcon.h"
```

C. SAMPLE PROGRAM

The following is an example of a C-language vector general graphics program. The actual picture produced by this program is illustrated in Figure A-6.

```
#include "/usr/lib/vgcon.h"

main()
{
 int i;
    sysinit();  //initailize the vector general

    //define coordinate system
    coordsys(2,-1.0,1.0,-1.0,1.0);

    drawele("box");
        setvector(VA,LN);
            move(-1.0,-1.0);
            line( 1.0, 1.0);
            line( 1.0, 1.0);
            line(-1.0, 1.0);
            line(-1.0,-1.0);
    endele();

    charele("name","%I BOX",SZ4,HOR,NSLNT,VGREL);

    drawele("zigzag");
        setvector(VA,LN);
            move(-0.5,-0.5)
        setvector(INCX,LN,.0308,NMG);
            move(.0308,.0308);
            for(i=0 ; i< 7 ; i++ )
            line(.0308,.0308);
    endele();

    //establish object-element relationship
    object(3,"bigbox","box","name","zigzag");
    object(3,"smallbox","box","name","zigzag");

    scale("bigbox",0.5);
    trans("bigbox",-0.5,-0.5);
    scale("smallbox",0.25);
    trans("smallbox",0.5,0.5);

    vgpicture();     //display picture
    sleep(30);  //display picture for 30 sec
    vgterm();   //terminate vector general process
}
```



Picture Displayed by Sample Program

FIGURE A-6

D. ERROR DIAGNOSTICS

All error messages will be printed on the PDP-11 terminal screen during the execution of a vector general graphics program. The error message will identify the routine that was executing when the error occurred and the error. The errors will be identified by a number. A user can identify the problem by referring to the listing of errors in this manual or by referencing the specific routine in Appendix B. Most errors will usually result in the termination of the routine being executed with control being returned to the user program. Every effort will be made to execute the program to completion, so at least some of the users data will be displayed on the vector general display screen. A description of each possible error is listed in Table-III.

TABLE-III
ERROR DIAGNOSTICS

Routine Error Numbers:

arc	1	intscale	12
blink	2	lightpen	13
charele	3	remove	17
circle	4	move	15
coordsys	5	object	16
copyele	6	remove	17
drawele	7	rotate	18
endele	8	scale	19
erase	9	setvector	20
indata	10	sysinit	21
intoffset	11	trans	22

Error Diagnostics:

E01 Element block error. The specified routine has been called outside of a drawele block. The instruction will be ignored and control will be returned to the user's program.

E02 Space allocated by the system for the building of elements has been exceeded. See the routine remove in order to free up unecessary element memory locations. The instruction executing when this error occurs will be ignored and control will be returned to the user's program.

E03 The value of the parameter dir was not C or CC. The routine will be ignored and no arc or circle will be drawn on the display.

E04 A character symbol included in an ASCII character string is undefined. The symbol will be ignored and control will be returned to the user's program.

E05 The named object does not exist, it has not been defined in a call to the routine object.

E06 The named element does not exist, it has not been defined in a call to either drawele or charele.

E07 An element name has been duplicated in a drawele or charele call. Each element structure created by drawele or charele must have a unique name.

E08 The total number of elements allowed by the system has been exceeded.

E09 The number of objects allowed by the system has been exceeded.

E11 The value of the parameter dim, specifying the desired number of dimensions for the user coordinate system is not a 2 or 3.

E14 The specified increment value falls outside of the defined coordinate system, the value will be ignored and control will be returned to the user's program.

E15 The x coordinate value, or the x increment value falls outside of the user's defined coordinate system. The instruction executing at the time of the error will be ignored and the move, line, arc or circle requested will not be drawn on the display screen.

E16 The y coordinate value, or the y increment value falls outside of the user's defined coordinate system. The instruction executing at the time of the error will be ignored and the move, line, arc or circle requested will not be drawn on the display screen.

E17 The z coordinate value, or the z increment value falls outside of the user's defined coordinate system. The instruction executing at the time of the error will be ignored and the move, line, arc or circle requested will not be drawn on the display screen.

E18 The value of the parameter num, specifying the number of elements being passed in this routine is not between 1 and 10.

E19 A circle or arc cannot be drawn by an incremental vector.

E20 The length of the user buffer specified for the routine indata is too small. The amount of data input from the keyboard exceeded the buffer size, characters received after the buffer boundry will be ignored.

E21 A named element within a parameter list is not associated with the specified object. The element was not linked to the named object in a call to the routine object. The routine executing at the time of this will stop and return control to the user's program.

APPENDIX B: USER INTERFACE ROUTINE DESCRIPTIONS

The description of the user interface routines appear in the same format as the routine descriptions in the UNIX Reference Manual at the Naval Postgraduate School Computer Laboratory.

NAME:

arc - draw an arc

SYNOPSIS:

```
arc(dir,centx [,centy] [,centz],endx [,endy] [,endz]);

dir: CC / 004 - counter clockwise
      C / 010 - clockwise

float centx [,centy] [,centz],endx [,endy] [,endz] ;
```

DESCRIPTION:

An arc is drawn from the present beam location, about the designated center point, to the user specified end point. If the center point is not an equal distance from the starting point of the arc and the end point, an arc will be drawn using one of the distances as the radius. The result, is a straight line drawn from the actual terminating point of the arc to the user specified end point.

The parameters passed for the center point and end point of the arc are either relative or absolute values. The number and value of these parameters are determined by the vector type selected in the previous call to setvector.

Arcs cannot be drawn by any of the four incremental vectors.

The z parameters are required only when the coordinate system is defined as three dimensional, and will be ignored if included when 2 dimensional is specified.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E03 - The draw direction was not defined as either C or CC.
Error - E15 - The x value was out of bounds.

Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.
Error - E19 - Illegal arc/circle instruction from an incremental vector.

ALSO SEE:

setvector - gives parameter requirements for each vector type.

NAME:

blink - blink the entire picture or the specified object or elements

SYNOPSIS:

action:

ON / 01 - blink
OFF / 00 - stop blinking action

Parameter values:

PIC / 00
OBJ / -1

To blink the entire picture:

```
blink(action,PIC);
```

To blink a specific object:

```
blink(action,OBJ,"objname");
```

To blink elements of a specific object:

```
blink(action,num,"objname","ele1",ele2",...);

int num;
```

DESCRIPTION:

The blink mode for the entire picture, a single object or any number of elements of a specific object, can be turned on or off.

Modifying the blink mode of an object affects all elements linked to that object.

num specifies the number of element names being passed in this routine, num can vary between 1 and 10.

DIAGNOSTICS:

All errors will be printed on the PDP-11 terminal

screen.

Error - E05 - Unknown object name.
Error - E06 - Unknown element name.
Error - E18 - Value num not between 1-10.

NAME:

charele - displays an ASCII character string
on the Vector General

SYNOPSIS:

```
charele("elename",charptr,size,wdir,slant,xpos,ypos);

float xpos, ypos;

size:
     SZ  /   00  - use previous character size
     SZ1 / 0100 - set size to 100 col. by 60 lines
     SZ2 / 0120 - set size to 81 col. by 41 lines
     SZ3 / 0140 - set size to 60 col. by 30 lines
     SZ4 / 0160 - set size to 32 col. by 16 lines

wdir:
     HOR /   00 - write characters horizontally
     VER / 0200 - write characters vertically

slant:
     SLNT  / 00 - slant characters
     NSLNT / 01 - do not slant characters

xpos:
     VGREL- to output characters relative to the
            present beam position the parameter xpos
            should be replaced with VGREL and
            parameter ypos should be omitted.
```

DESCRIPTION:

This routine displays the ASCII character string, specified by charptr, on the vector general screen. The characters can be output by specifying the desired coordinate starting point (xpos,ypos) or by outputing the string relative to the present beam location. The symbols available include formatting symbols and an extended character set. These can be included in any character string by including the proper identifying symbols. The character string formatting symbols are:

KEYBOARD SYMBOL	FORMAT
%@	backspace,moves back one character space
%A	line feed, position center screen
%B	line feed
%C	position at top,center of screen
%D	position at top,left corner of screen
%E	carriage control, line feed
%F	ignored
%G	ignored
%H	special character
%I	neg. line feed,moves up one line
%J	decreases current character size by one
%K	increases current character size by one

Due to the extended character set available on the vector general, all special character symbols are preceded by a percent sign. In order to have a percent sign appear on the screen, two percent signs (%%) must appear in the character string. A list of the special characters available are listed on the next page under Extended Character Set.

DIAGNOSTICS:

All errors will be printed on the PDP-11 terminal screen.

Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E04 - Incorrect ASCII character symbol found in character string.
Error - E09 - Maximum number of elements per picture exceeded.

Extended Character Set

KEYBOARD SYMBOL	CHARACTER
%space	□
%!	↓
%"	‖
%#	◯
%$	&
%H	∨
%&	∫
%'	√
%(	⊂
%)	⊃
%*	10
%+	÷
%,	≦
%-	≡
%.	≧
%/	⌐
%0	∘
%1	↑
%2	↙
%3	▢
%4	≠
%5	∧

charele

%6	∂
%7	∠
%8	∪
%9	∩
%:	•
%;	×
%<	←
%M	⅀
%N	ℵ
%O	Ω
%P	∏
%Q	∩
%R	∮
%S	Σ
%T	⊖
%U	∣
%V	√
%W	∧
%X	↑
%Y	⌐
%Z	—
%[	L
%	⇒
%]	⊢
%↑	_ *
%←	∘ *

%`	⌐
%a	α
%b	β
%c	∇
%d	δ
%e	ϵ
%f	ϕ
%g	γ
%h	⊥
%i	ι
%j	↘
%k	↗
%l	λ
%m	μ
%n	ν
%o	ω
%p	π
%q	Б
%r	ρ
%s	σ
%t	τ
%u	ↄ
%v	○
%w	↙
%x	!
%y	∠

charele

%z	
%{	
%!	
%}	
%¬	
%?	

NAME:

circle - draw a circle

SYNOPSIS:

```
circle(dir,centx [,centy] [,centz] )

dir: CC / 004 - counter clockwise
      C / 010 - clockwise

float centx [,centy] [,centz] ;
```

DESCRIPTION:

A circle is drawn from the present beam location about the indicated center point.

The parameters passed for the center point of the circle are either relative or absolute values. The number and value of these parameters are determined by the vector type selected in the previous call to setvector.

Circles cannot be drawn by any of the four incremental vectors.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

```
Error - E01 - This routine has been called outside
              of a draw element block.
Error - E02 - The space allocated for the building
              of elements has been exceeded.
Error - E03 - The draw direction was not defined
              as either C or CC.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.
Error - E19 - Illegal arc/circle instruction
              from an incremental vector.
```

ALSO SEE:

setvector - gives parameter requirements for each vector type.

NAME:

coordsys - define user coordinate system

SYNOPSIS:

```
coordsys(dim,minx,maxx,miny,maxy [,minz,maxz] );

int dim;

float minx,maxx,miny,maxy [,minz,maxz] ;
```

DESCRIPTION:

Defines a two or three dimensional cartesian coordinate system, of any scale, for the user. The parameter dim specifies the number of dimensions required. If dim is 2 then only the range of x and y need to be specified.

minx must be strictly less than maxx.

miny must be strictly less than maxy.

minz must be strictly less than maxz.

All subsequent user coordinate values are interpreted according to this user defined coordinate system.

If coordsys is not called by the user the default values will be taken. The default coordinate system is three dimensional with x, y, z ranging from -100.0 to 100.0. All coordinate values received will be interpreted according to these default values unless the coordinate system is redefined by the user.

DIAGNOSTICS:

Any values which fall outside of the defined coordinate system will be ignored and an error message specifying which coordinate values were out of range will be printed on the PDP-11 terminal screen.

NAME:

copyele - specify additional names to be used in referencing a specific element block

SYNOPSIS:

```
copyele(num,"elename","name1","name2",...);

int num;
```

DESCRIPTION:

This routine allows the user to give several unique names to a specific element block. In this manner, a user can link one element to an object several times and can uniquely reference each occurrence of the element within the object.

num specifies the number of element names being passed in this routine, num can vary between 1 and 10.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E06 - Unknown element name.
Error - E07 - Duplicate element name.
Error - E18 - Value num not between 1-10.

ALSO SEE:

drawele, object

NAME:

drawele - start a draw element block

SYNOPSIS:

```
drawele("elename");
```

DESCRIPTION:

This routine specifies the beginning of a draw element block. It associates a unique name with the group of draw instructions that fall in between this call and a call to endele. The resulting picture segment will then be referenced by the name specified, as the element name, in this routine.

A user may want to repeat a specific element block structure several times within one object. Instead of specifying several element blocks which describe the same structure a user can indicate a group of names which refer to one specific element block. These unique names each referring to the same structure can then be linked to an object, and each can be uniquely referenced. This association of several names with one element can be accomplished by the routine copyele.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E07 - Duplicate element name.

ALSO SEE:

copyele, object

NAME:

endele - end of the current element block

SYNOPSIS:

```
endele();
```

DESCRIPTION:

Specifies the termination of a list of draw instructions describing a specific element. The picture segment described by this group of draw instructions, that fall between a drawele and endele call, will be referenced by name specified in the drawele.

A new element block cannot begin until the previous block has been properly terminated by a call to endele.

DIAGNOSTICS:

This routine must be called to properly end a draw element block. If a block is not properly terminated prior to the beginning of a new element block, all drawele calls will be ignored and any draw instructions that follow will be associated with the element block that has not been terminated.

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.

NAME:

erase - erase the specified portions of the picture

SYNOPSIS:

Parameter values:

```
PIC / 00
OBJ / -1
```

To erase the entire picture:

```
erase(PIC);
```

To erase a specific object:

```
erase(OBJ,"objname");
```

To erase elements of a specific object:

```
erase(num,"objname","ele1","ele2",..);
int num;
```

DESCRIPTION:

The entire picture, specified object or the listed elements of a specific object, will be erased from the vector general display screen. The elements still exist. To redisplay any portion of the erased picture, the appropriate element-object linking must again be done by the user.

num specifies the number of element names being passed in this routine.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

```
Error - E05 - Unknown object name.
Error - E06 - Unknown element name.
Error - E18 - Value num not between 1-10.
Error - E21 - Element not associated with
              the named object.
```

ALSO SEE:

remove

NAME:

errormsg - print error messages on PDP-11 terminal screen

SYNOPSIS:

```
errormsg(action);

action:
        ON  / 01 - print errors on PDP-11 terminal
        OFF / 00 - stop printing of error messages
```

DESCRIPTION:

All error messages will automatically be printed on the PDP-11 terminal screen. The user can control the printing of error messages during any portion of a program by calling this routine.

NAME:

indata - input data from the vector general keyboard

SYNOPSIS:

```
indata(buffer,length,xpos,ypos);

char *buffer;

int length;

float xpos,ypos;
```

DESCRIPTION:

This routine allows the user to enter data from the vector general keyboard onto the vector general display screen beginning at the position indicated by xpos, ypos. The data, ASCII character codes, will also be placed in the user buffer specified by buffer. The length of the user buffer must be specifed by the parameter length. A cursor will be displayed at the initial position and all data will remain on the display screen until a carriage return is entered by the user. All data placed in the user's buffer will remain unaltered.

A listing of the ASCII character code for every keyboard entry is listed on the next page.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E20 - User buffer dimensioned to small.

OCTAL CODE	CHARACTER	GENERATED SYMBOL	KEYBOARD SYMBOL
000	NULL	(ignored)	@ ctrl
001	SO	erase last char.	A ctrl
002	STX	(ignored)	B ctrl
003	ETX	(ignored)	C ctrl
004	EOT	(ignored)	D ctrl
005	ENQ	(ignored)	E ctrl
006	ACK	(ignored)	F ctrl
007	BEL	(ignored)	G ctrl
010	BS		BS
011	HT	(LF, cent)	I ctrl
012	LF		LF
013	VT	(top, cent)	K ctrl
014	FF	(top, left)	L ctrl
015	NL	(CR, LF)	CR
016	SE	(ignored)	N ctrl
017	SI	(ignored)	O ctrl
020	DLE	(clear queue)	P ctrl
021	DC1	(-LF)	Q ctrl
022	DC2	(-SZ)	R ctrl
023	DC3	(+SZ)	S ctrl
024	DC4	(terminate)	T ctrl
025	NAK	(ignored)	U ctrl
026	SYN	(ignored)	V ctrl
027	ETB	(ignored)	W ctrl
030	CAN	(ignored)	X ctrl
031	EM	(ignored)	Y ctrl
032	SUB	(ignored)	Z ctrl
033	ESC	(escape)	[ctrl
034	FS	(ignored)	\ ctrl
035	GS	(ignored)	] ctrl
036	RS	(ignored)	^ ctrl
037	US	(ignored)	
040	Space		sp bar
041	!		1 shift
042	"		2 shift
043	#		3 shift
044	$		4 shift
045	%		5 shift
046	&		6 shift
047	'		7 shift
050	(		8 shift
051	)		9 shift
052	*		: shift
053	+		; shift
054	,		,
055	-		-
056	.		.
057	/		/
060	0		0

indata

061	1	1
062	2	2
063	3	3
064	4	4
065	5	5
066	6	6
067	7	7
070	8	8
071	9	9
072	:	:
073	;	;
074	<	, shift
075	=	- shift
076	>	. shift
077	?	?
100	@	@
101	A	A shift
102	B	B shift
103	C	C shift
104	D	D shift
105	E	E shift
106	F	F shift
107	G	G shift
110	H	H shift
111	I	I shift
112	J	J shift
113	K	K shift
114	L	L shift
115	M	M shift
116	N	N shift
117	O	O shift
120	P	P shift
121	Q	Q shift
122	R	R shift
123	S	S shift
124	T	T shift
125	U	U shift
126	V	V shift
127	W	W shift
130	X	X shift
131	Y	Y shift
132	Z	Z shift
133	[	[
134	\	\
135	]	]
136	^	^
137	←(sub,superscript)	_
140	`	@ shift
141	a	A
142	b	B
143	c	C
144	d	D

145	e		E
146	f		F
147	g		G
150	h		H
151	i		I
152	j		J
153	k		K
154	l		L
155	m		M
156	n		N
157	o		O
160	p		P
161	q		Q
162	r		R
163	s		S
164	t		T
165	u		U
166	v		V
167	w		W
170	x		X
171	y		Y
172	z		Z
173	{		] shift
174	¦		\ shift
175	}		[shift
176	~		^ shift
177	del		DEL
240	□		space spec
241	↓		1 shift spec
242	‖		2 shift spec
243	◯		3 shift spec
244	£		4 shift spec
245	∨		5 shift spec
246	∫		6 shift spec
247	≺		7 shift spec
250	⊂		8 shift spec
251	⊃		9 shift spec
252	10	(subscript)	: shift spec
253	÷		; shift spec
254	≦		, spec
255	≧		- spec
256	≠		. spec
257	¬		/ spec
260	$_0$		0 spec
261	↑		1 spec
262	¥		2 spec
263	□		3 spec
264	¢		4 spec
265	>	(centered)	5 spec
266	∂		6 spec
267	∠		7 spec
270	∪		8 spec

indata

271		9 spec
272	(centered)	: spec
273		; spec
274		, shift spec
315		M shift spec
316		N shift spec
317		O shift spec
320		P shift spec
321		Q shift spec
322		R shift spec
323		S shift spec
324		T shift spec
325		U shift spec
326		V shift spec
327		W shift spec
330		X shift spec
331		Y shift spec
332		Z shift spec
333		[shift spec
334		spec
335		1 shift spec
336		^ spec
337		_ spec
340	(blinking)	@ shift spec
341		A shift spec
342		B shift spec
343		C shift spec
344		D shift spec
345		E shift spec
346		F shift spec
347		G shift spec
350		H shift spec
351		I shift spec
352		J shift spec
353		K shift spec
354		L shift spec
355		M shift spec
356		N shift spec
357		O shift spec
360		P shift spec
361		Q shift spec
362		R shift spec
363		S shift spec
364		T shift spec
365		U shift spec
366		V shift spec
367		W shift spec
370		X shift spec
371		Y shift spec
372		Z shift spec
373		[spec shift
374		spec shift

indata

375	⊣	] spec shift
376	~•	^ spec shift
377	▥	DEL spec

NAME:

intoffset - object intensity offset

SYNOPSIS:

```
intoffset("objname",val);

float val;
```

DESCRIPTION:

The intensity range of the specified object is determined by the parameter val. If val is one, the maximum intensity range is achieved. If the value is zero, the intensity is constant and the image has no depth-cueing.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.

ALSO SEE:

intscale

NAME:

intscale - modify object intensity scale

SYNOPSIS:

```
intscale("objname",val);

float val;
```

DESCRIPTION:

The intensity range of an object, which gives a three dimensional object its depth cueing, is determined by the parameter val. The range of val is from zero to one. If val is one the maximum intensity range is obtained, if it is zero the object has no depth cueing.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.

ALSO SEE:

intoffset

NAME:

lghtpen - set lightpen hookability of
the picture,object or elements

SYNOPSIS:

action:

ON / 01 - Set light pen hookability
OFF / 00 - Clear light pen hookability

Parameter values:

PIC / 00
OBJ / -1

To set hookability of the entire picture:

```
lghtpen(action,PIC);
```

To set hookability of an object:

```
lghtpen(action,OBJ,"objname");
```

To set hookability of elements of a specific object:

```
lghtpen(action,num,"objname","ele1","ele2",...);

int num;
```

DESCRIPTION:

The user can specify which picture segments will be light pen hookable. These elements designated as light pen hookable will be effected by light pen interaction with the vector general display screen.

num specifies the number of element names being passed in this routine, num can vary between 1 and 10.

DIAGNOSTICS:

All errors will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.

Error - E06 - Unknown element name.
Error - E18 - Value num not between 1-10.
Error - E21 - Element not associated with the named object.

AD-A032 324 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 9/2
THE DESIGN AND IMPLEMENTATION OF A GENERAL PURPOSE INTERACTIVE --ETC(U)
SEP 76 B J STANKOWSKI

UNCLASSIFIED NL

2 OF 2
ADA032324

END

DATE
FILMED
1 -77

NAME:

line - draw a line

SYNOPSIS:

```
line(parx [,pary] [,parz] );

float parx [,pary] [,parz] ;
```

DESCRIPTION:

A line is drawn to the location specified by the x,y,z coordinate values. The values may specify actual absolute coordinate points or an incremental value which will be added to or subtracted from the previous x,y,z coordinate point. The number of parameters and their values are determined by the vector type specified in the preceding call to setvector.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.

ALSO SEE:

setvector - gives parameter requirements for each vector type.

NAME:

move - move to the specified location

SYNOPSIS:

```
move(parx [,pary] [,parz] );

float parx [,pary] [,parz] ;
```

DESCRIPTION:

The beam is moved to the location specified by the x,y,z coordinate values. The values may specify actual absolute coordinate points or an incremental value which will be added to or subtracted form the previous x,y,z coordinate values. The number of parameters and their values are determined by the vector type specified in the preceding call to setvector.

DIAGNOSTIC:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.

ALSO SEE:

setvector - gives parameter requirements for each vector type.

NAME:

object - link elements to specified object name

SYNOPSIS:

```
object(num,"objname","ele1","ele2",....);

int num;
```

DESCRIPTION:

Associates with the named object each of the listed elements. This picture segment, or element grouping will be referenced by the object name specified in this routine.

Every element must be linked to at least one object inorder for it to be displayed on the vector general screen. A user can link one or more element names to an object. Elements can be linked to an object by one or several calls to this routine. An element can be linked to several different objects, or one element may be linked several times to the same object.

If an object or element has been erased from the display screen, a user can redisplay the desired object or elements by again establishing the desired object-element association.

num specifies the number of element names being passed in this routine, num can vary between 1 and 10.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

```
Error - E05 - Unknown object name.
Error - E06 - Unknown element name.
Error - E08 - Number of elements allowed
              per picture exceeded.
Error - E09 - Maximum number of objects
              allowed per picture (10) exceeded.
```

ALSO SEE:

drawele, copyele

NAME:

remove - erase the indicated portion of the picture and remove the associated elements from memory

SYNOPSIS:

Parameter values:

PIC / 00

To remove the entire picture:

```
remove(PIC);
```

To remove element structures :

```
remove(num,"ele1","ele2",..);
int num;
```

DESCRIPTION:

The entire picture, or every occurrence of the named elements are erased from the vector general display screen. Each name element structure will be removed from memory.

To redisplay any portion of the removed picture requires that the user rebuild each element and relink it to the appropriate objects.

num specifies the number of element names being passed in this routine.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.
Error - E06 - Unknown element name.
Error - E18 - Value num not between 1-10.
Error - E21 - Element not associated with the named object.

ALSO SEE:

erase

NAME:

rotate - rotate an object

SYNOPSIS:

```
rotate("objname",xradians,yradians [,zradians] );

float xradians,yradians [,zradians] ;
```

DESCRIPTION:

The named object will be rotated about the x, y, z coordinate axis. The parameters, given in radian measure, specify the degree of rotation desired about each axis.

The z parameter is required only when the coordinate system is defined as three dimensional and will be ignored otherwise.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.

NAME:

scale - modify object scale

SYNOPSIS:

```
scale("objname", scale);

float scale;
```

DESCRIPTION:

All elements associated with the named object are scaled by the value scale at display time. The range of scale is from zero to one. If not established by a user this value automatically defaults to one, which is the maximum picture scale.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.

NAME:

setvector - specify the vector type and vector mode

SYNOPSIS:

```
setvector(vtype,vmode [,inc][,scale] );

vtype:
        VR  - vector relative
        VRX - vector relative auto-increment x
        VRY - vector relative auto-increment y
        VRZ - vector relative auto-increment z
        VA  - vector absolute
        VAX - vector absolute auto-increment x
        VAY - vector absolute auto-increment y
        VAZ - vector absolute auto-increment z
        INC2 - vector incremental 2-dimensional
        INCX - vector incremental auto-increment x
        INCY - vector incremental auto-increment y
        INC3 - vector incremental 3-dimensional

vmode:
        LN  /  00 - line
        DSH / 020 - dashed line
        DOT / 040 - dotted line
        PNT / 060 - end point
        DD  /0120 - dash-dot-dash line
        DDD /0140 - dash-dot-dash line

scale:

        MG  / 000 - add the  coordinate increments
                    to the high order bits of the
                    specified register.
        NMG /0200 - add the coordinate increments
                    to the low order bits of the
                    specified register.


float inc;
```

DESCRIPTION:

This routine specifies which one of the 12 vector types is to be drawn in the line, move, circle and arc instructions which follow. It also specifies the vector mode (i.e., line, dotted, dashed, etc.), and if required, the increment value for auto-increment vectors and the scale factor for incremental vectors.

The coordinate values passed in the following draw instructions are dependent on the vector type selected in this routine.

This must follow a drawele call and preced any draw instructions. This routine can be called any number of times within a draw element block.

The bracketed z parameters are required only when the coordinate system has been defined as three dimensional.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.
Error - E19 - Illegal arc/circle instruction from an incremental vector.

NAME:

VA - vector absolute

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(VA,vmode);
    move(coordx,coordy [,coordz] );
    line(coordx,coordy [,coordz] );
    circle(dir,centx,centy [,centz] );
    arc(dir,centx,centy [,centz],endx,endy [,endz]);
    .
    .
endele();

float coordx,coordy [,coordz] ;
float centx,centy [,centz],endx,endy [,endz];
```

DESCRIPTION:

The x,y,z coordinates of absolute vectors are specified with respect to the origin, or zero position of the user defined coordinate system. Each x, y, z coordinate value references a unique point on the display screen.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.

NAME:

VAX - vector absolute auto-increment x

SYNOPSIS:

```
drawele("elename");
     .
     .
     setvector(VAX,vmode,xinc);
     move(coordy [,coordz] );
     move(coordy [,coordz] );
     circle(dir,centy [,centz] );
     arc(dir,centy [,centz],endy [,endz]);
     .
     .
endele();

float xinc, coordy [,coordz] ;
float centy [,centz],endy [,endz];
```

DESCRIPTION:

The vector absolute auto-increment x causes the initial x coordinate value to be incremented or decremented by the value specified by xinc. With every move, line, arc, or circle command the specified y and z coordinate values are updated while the x coordinate value is stepped by the constant value xinc. For example, the point(x, y, z) becomes point(x+xinc, coordy, coordz).

The value of xinc cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.

NAME:

VAY - vector absolute auto-increment y

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(VAY,vmode,yinc);
    move(coordx [,coordz] );
    line(coordx [,coordz] );
    circle(dir,centx [,centz] );
    arc(dir,centx [,centz],endx [,endz]);
    .
    .
endele();

float yinc,coordx [,coordz] ;
float centx [,centz],endx [,endz];
```

DESCRIPTION:

The vector absolute auto-increment y causes the initial y coordinate value to be incremented or decremented by the value specified by yinc. With every move, line, arc or circle command the specified x and z coordinate values are updated while the y coordinate value is stepped by the constant value yinc. For example, the point(x,y,z) becomes point(coordx,y+yinc, coordz).

The value of yinc cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E15 - The x value was out of bounds.
Error - E17 - The z value was out of bounds.

NAME:

VAZ - vector absolute auto-increment z

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(VAZ,vmode,zinc);
    move(coordx,coordy);
    line(coordx,coordy);
    circle(dir,centx,centy);
    arc(dir,centx,centy,endx,endy);
    .
    .
endele();

float zinc,coordx,coordy;
float centx,centy,endx,endy;
```

DESCRIPTION:

The vector absolute auto-increment z causes the initial z coordinate value to be incremented or decremented by the value specified by zinc. With every move, line, arc or circle command the specified x and y coordinate values are updated while the z coordinate value is stepped by the constant value zinc. For example the point(x,y,z) becomes point(coordx,coordy,z+zinc).

The value of zinc cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.

NAME: VR - vector relative

SYNOPSIS:

```
drawele("elename");
     .
     .
     setvector(VR,vmode);
     move(deltax,deltay [,deltaz] );
     line(deltax,deltay [,deltaz] );
     circle(dir,centx,centy,[centz]);
     arc(dir,centx,centy[,centz],endx,endy[,endz]);
     .
     .
endele();

float deltax,deltay [,deltaz] ;
float centx,centy [,centz],endx,endy [,endz];
```

DESCRIPTION:

Relative vectors specify a increment value that is to be added to or subtracted from the initial x,y,z absolute coordinate values. For example, the point(x,y,z) becomes point(x+deltax, y+deltay,z+deltaz).

The parameter values cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.

NAME:

VRX - vector relative auto-increment x

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(VRX,vmode,xinc)
    move(deltay [,deltaz] );
    line(deltay [,deltaz] );
    circle(dir,centy [,centz]);
    arc(dir,centy [,centz],endy [,endz]);
    .
    .
endele();

float xinc,deltay [,deltaz] ;
float centy [,centz],endy [,endz];
```

DESCRIPTION:

A vector relative auto-increment x causes the initial x coordinate value to be incremented or decremented by the constant value xinc. With every move, line, arc or circle instruction that follows, the y and z coordinate values will be updated, while x is stepped by the value xinc. For example, the point(x,y,z) becomes point(x+xinc,y+deltay,z+deltaz).

The parameter values cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.

NAME:

VRY - vector relative auto-increment y

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(VRY,vmode,yinc)
    move(deltax [,deltaz] );
    line(deltax [,deltaz] );
    circle(dir,centx [,centz]);
    arc(dir,centx [,centz],endx [,endz]);
    .
    .
endele();

float yinc,deltax [,deltaz] ;
float centx [,centz],endx [,endz];
```

DESCRIPTION:

A vector relative auto-increment y causes the initial y coordinate value to be incremented or decremented by the constant value yinc. With every move, line, arc or circle instruction that follows, the x and z coordinate values will be updated, while y is stepped by the value yinc. For example, the point(x,y,z) becomes point(x+deltax,y+yinc,z+deltaz).

The parameter values cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E15 - The x value was out of bounds.
Error - E17 - The z value was out of bounds.

NAME:

VRZ - vector relative auto-increment z

SYNOPSIS:

```
drawele("elename");
     .
     .
     setvector(VRZ,vmode,zinc)
     move(deltax, deltay);
     line(deltax ,deltay);
     circle(dir,centx,centy);
     arc(dir,centx,centy,endx,endy);
     .
     .
endele();

float zinc,deltax,deltay;
float centx,centy,endx,endy;
```

DESCRIPTION:

A vector relative auto-increment z causes the initial z coordinate value to be incremented or decremented by the constant value zinc. With every move, line, arc or circle instruction that follows, the x and y coordinate values will be updated, while z is stepped by the value zinc. For example, the point(x,y,z) becomes point(x+deltax,y+deltay,z+incz).

The parameter values cannot exceed the range of the defined coordinate system.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.

NAME:

INC2 - incremental vector , two dimensional

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(INC2,vmode,scale);
    move(deltax,deltay);
    line(deltax,deltay);
    .
    .
endele();

float deltax,deltay;
```

scale:
MG - add the coordinate increments to the 7 high order bits of the specified register.
NMG- add the coordinate increments to the 7 low order bits of the specified register.

DESCRIPTION:

A two dimensional relative vector, that halves vector storage requirements and doubles the data rate. This vector type should be used when element storage space is critical. This reduced storage requirement results from a limitation on the maximum size of each relative increment. The values passed in line or move instructions are limited to approximately 3% of the user minimum and maximum coordinate ranges. The parameter, scale, specifies if this increment value should be added to the 7 high order bits of the specified coordinate or to the 7 low order bits. This scale factor then determines if the increments will be applied over a fine(NMG) or coarse(MG) grid.

Arcs and circles cannot be drawn with this vector type.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside

of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E19 - Illegal arc/circle instruction from an incremental vector.

NAME:

INCX - incremental vector ,two dimensional
auto-increment x

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(INCX,vmode,xinc,scale);
    move(deltay,deltay);
    line(deltay,deltay);
    .
    .
endele();
```

```
float xinc,deltay;
```

scale:
MG - add the coordinate increments to the 7 high order bits of the specified register.
NMG- add the coordinate increments to the 7 low order bits of the specified register.

DESCRIPTION:

A two dimensional relative vector, that halves vector storage requirements and doubles the data rate. This vector type should be used when element storage space is critical. This reduced storage requirement results from a limitation on the maximum size of each relative increment. The values passed in line or move instructions are limited to approximately 3% of the user minimum and maximum coordinate ranges. The parameter, scale, specifies if this increment value should be added to the 7 high order bits of the specified coordinate or to the 7 low order bits. This scale factor then determines if the increments will be applied over a fine(NMG) or coarse(MG) grid.

This incremental, auto increment vector causes the x coordinate value to be incremented or decremented by the constant value xinc, while the y coordinate value is increment by small relative values.

Arcs and circles cannot be drawn with this vector type.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E19 - Illegal arc/circle instruction from an incremental vector.

NAME:

INCY - incremental vector ,two dimensional
auto-increment y

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(INCY,vmode,yinc,scale);
    move(deltax,deltax);
    line(deltax,deltax);
    .
    .
endele();

float yinc,deltax;
```

scale:
MG - add the coordinate increments to the 7 high order bits of the specified register.
NMG- add the coordinate increments to the 7 low order bits of the specified register.

DESCRIPTION:

A two dimensional relative vector, that halves vector storage requirements and doubles the data rate. This vector type should be used when element storage space is critical. This reduced storage requirement results from a limitation on the maximum size of each relative increment. The values passed in line or move instructions are limited to approximately 3% of the user minimum and maximum coordinate ranges. The parameter, scale, specifies if this increment value should be added to the 7 high order bits of the specified coordinate or to the 7 low order bits. This scale factor then determines if the increments will be applied over a fine(NMG) or coarse(MG) grid.

This incremental, auto increment vector causes the y coordinate value to be incremented or decremented by the constant value xinc, while the y coordinate value is increment by small relative values.

Arcs and circles cannot be drawn with this vector type.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E14 - The increment value was out of bounds.
Error - E15 - The x value was out of bounds.
Error - E19 - Illegal arc/circle instruction from an incremental vector.

NAME:

INC3 - incremental vector ,three dimensional

SYNOPSIS:

```
drawele("elename");
    .
    .
    setvector(INCX,vmode,scale);
    move(deltax,deltay,deltaz);
    line(deltax,deltay,deltaz);
    .
    .
endele();
```

```
float deltax,deltay,deltaz;
```

scale:
MG - add the coordinate increments to the 7 high order bits of the specified register.
NMG- add the coordinate increments to the 7 low order bits of the specified register.

DESCRIPTION:

A three dimensional relative vector, that halves vector storage requirements and doubles the data rate. This vector type should be used when element storage space is critical. This reduced storage requirement results from a limitation on the maximum size of each relative increment. The values passed in line or move instructions are limited to approximately 3% of the user minimum and maximum coordinate ranges. The paramater, scale, specifies if this increment value should be added to the 7 high order bits of the specified coordinate or to the 7 low order bits. This scale factor then determines if the increments will be applied over a fine(NMG) or coarse(MG) grid.

Arcs and circles cannot be drawn with this vector type.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E01 - This routine has been called outside of a draw element block.
Error - E02 - The space allocated for the building of elements has been exceeded.
Error - E13 - Vector type undefined.
Error - E15 - The x value was out of bounds.
Error - E16 - The y value was out of bounds.
Error - E17 - The z value was out of bounds.
Error - E19 - Illegal arc/circle instruction from an incremental vector.

NAME:

sysinit - vector general display initialization

SYNOPSIS:

```
sysinit();
```

DESCRIPTION:

This routine establishes a link with the vector general, initializes its display system and sets all the user default parameters.

This routine must be called before any other display instructions.

DIAGNOSTICS:

If the initialization cannot be properly completed, an error message will be printed on the PDP-11 terminal and the process will be terminated. This error could occur because the vector general is being accessed by a another user.

NAME:

trans - translate object

SYNOPSIS:

```
trans("objname",x,y [,z] );

float x,y [,z] ;
```

DESCRIPTION:

All of the elements associated with the named object are translated by the amount x, y, z at display time. Each coordinate point(X,Y,Z) of the object becomes point(X+x,Y+y,Z+z) at display time.

The z parameter is required only when the coordinate system is defined as three dimensional, and will be ignored if included.

DIAGNOSTICS:

All error messages will be printed on the PDP-11 terminal screen.

Error - E05 - Unknown object name.

BIBLIOGRAPHY

1. Graphics Display System Reference Manual, Vector General Inc., Woodland Hill, California, August 1974.

2. Ritchie, D. M., C Reference Manual, Bell Telephone Laboratories, Murray Hill, New Jersey, 1975.

3. Graphics Display System Technical Manual, Vector General Inc., Woodland Hill, California, June 1974.

4. Ritchie, D. M. and Thompson, K., The UNIX Time-Sharing System, Bell Telephone Laboratories, Murray Hill, New Jersey, 1974.

5. Naval Postgraduate School Technical Report NP572Rr76031, Design Manual for the Vector General Interactive Display Unit, by L. A. Thorpe and G. M. Raetz, March 1976.

6. Naval Postgraduate School Technical Report NP572Rr76031, User Manual for the Vector General Display Unit, by L. A. Thorpe and G. M. Raetz, March 1976.

7. Newman, W. M., Display Procedures, Communications of the ACM, p651-660, v.4, n10, October 1968.

8. Newman, W. M., and Sproull, R. F., Principles of Interactive Computer Graphics, McGraw-Hill, 1973.

9. Kulsrud, H. E., A General-purpose Graphics Language, Communication of the ACM, p247-254, v.4, no.11, April 1968.

10. Johnson, C. I., Principles of Interactive Systems, IBM Systems Journal, p147-168, v.7, no.3-4, 1968.

11. Lucido, A. P., Software Systems for Computer Graphics, p23-32, v.9, no.8, August 1976.

12. Newman, W. M, and Sproull, R. F., An Approach to Graphic Systems Design, Proceedings of the IEEE, p471-483, v.62, no., April 1974.

13. Foley, J. D., and Wallace V. L., The Art of Natural Graphic Man-Machine Conversation, Proceedings of the IEEE, p462-470, v.62, no.4, April 1974.

14. Foley, J. D., Picture Naming and Modification: An Overview, SIGPLAN Notices, p49-53, v.2, no.6, June 1976.

15. Wirth, N., Algorithms + Data Structures = Programs, Prentice-Hall, 1976.

16. Pratt, T. W., Programming Languages: Design and Implementation, Prentice-Hall, 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexanderia, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Computer Science Department Naval Postgraduate School Monterey, California 93940	1
4.	LTJG Gary M. Raetz, USN, Code 52Rr Computer Science Department Naval Postgraduate School Monterey, California 93940	1
5.	Professor George A. Rahe, code 52Ra Computer Science Department Naval Postgraduate School Monterey, California 93940	1
6.	LT Barbara J. Stankowski, USN 423 Martin Terrace State College, Pennsylvania 16801	1